The posterior-Viterbi: a new decoding algorithm for hidden Markov models

Piero Fariselli*, Pier Luigi Martelli, and Rita Casadio

Department of Biology, University of Bologna
via Irnerio 42, 40126 Bologna, Italy
Tel: +39-051-2091280 Fax: +39-051-242576

e-mails: piero.fariselli@unibo.it,
gigi@biocomp.unibo.it,
casadio@alma.unibo.it
∗ To whom correspondence should be addressed

ABSTRACT

Background: Hidden Markov models (HMM) are powerful machine learning tools successfully applied to problems of computational Molecular Biology. In a predictive task, the HMM is endowed with a *decoding algorithm* in order to assign the most probable state path, and in turn the class labeling, to an unknown sequence. The Viterbi and the posterior decoding algorithms are the most common. The former is very efficient when one path dominates, while the latter, even though does not guarantee to preserve the automaton grammar, is more effective when several concurring paths have similar probabilities. A third good alternative is 1-best, which was shown to perform equal or better than Viterbi.
Results: In this paper we introduce the posterior-Viterbi (PV) a new decoding which combines the posterior and Viterbi algorithms. PV is a two step process: first the posterior probability of each state is computed and then the best posterior allowed path through the model is evaluated by a Viterbi algorithm.
Conclusions: We show that PV decoding performs better than other algorithms first on toy models and then on the computational biological problem of the prediction of the topology of beta-barrel membrane proteins.
Contacts: piero.fariselli@unibo.it

Background

Machine learning approaches have been shown to be very profitable in the field of computational Molecular Biology [3]. Among them hidden Markov models (HMMs) have been proven to be especially successful when in the problem at hand regular grammar-like structures can be detected [3, 7]. HMMs were developed for alignments [11, 4], pattern detection [15, 5] and also for predictions, as in the case of the topology of all-alpha and all-beta membrane proteins [18, 14, 16, 17, 10, 19, 2, 6].

When HMMs are implemented for predicting a given feature, a *decoding* algorithm is needed. With decoding we refer to the assignment of a path through the HMM states (which is the best under *a suitable measure*) given an observed sequence O. In this way, we can also assign a class label to each sequence element of the emitting state [3, 7]. More generally, as stated in [13], the *decoding is the prediction of the labels of an unknown path.* Labeling is routinely the only relevant biological property associated to the observed sequence; the states themselves may not represent a significant piece of information, since they basically define the automaton grammar.

The most famous decoding procedure is the Viterbi algorithm, which finds the most probable allowed path through the HMM model. Viterbi decoding is particularly effective when there is a *single best path* among others much less probable. When several paths have similar probabilities, the posterior decoding or the 1-best algorithms are more convenient [13]. The posterior decoding assigns the state path on the basis of the posterior probability, although the selected path might be not allowed. For this reason, in order to recast the automaton constraints, a post-processing algorithm was applied to the posterior decoding [8].

In this paper we address the problem of preserving the automaton grammar and concomitantly exploiting the posterior probabilities, without the need of the post-processing algorithm [8, 16]. Prompted by this, we design a new decoding algorithm, the *posterior-Viterbi decoding* (PV), which *preserves the automaton grammars and at the same time exploits the posterior pobabilities.* We show that PV performs better than the other algorithms when we test it on toy models and on the problem of the prediction of the topology of beta-barrel membrane proteins.

Methods

The hidden Markov model definitions

For sake of clarity and compactness, in what follows we make use of explicit *BEGIN* and *END* states and we do not treat the case of the *silent* (*null*) states. Their inclusion in the algorithms is only a technical matter and can be done following the prescriptions indicated in [3, 7].

An observed sequence of length L is indicated as O (=$O_1 ... O_L$) both for a single-symbol-sequence (as in the standard HMMs) or for a vector-sequence as described before [16]. $label(s)$ indicates the label associated to the state s, while Λ (=$\Lambda_i, \ldots \Lambda_L$) is the list of the labels associated to each sequence position i obtained after the application of a decoding algorithm. Depending on the problem at hand, the labels may identify transmembrane regions, loops, secondary structures of proteins, coding/non coding regions, intergenic regions, etc. A HMM consisting of N states is therefore defined by three probability distributions

Starting probabilities:

$$a_{BEGIN,k} = P(k|BEGIN) \tag{1}$$

Transition probabilities:

$$a_{k,s} = P(k|s) \tag{2}$$

Emission probabilities:

$$e_k(O_i) = P(O_i|k) \tag{3}$$

The forward probability is

$$f_k(i) = P(O_1, O_2 \ldots O_i, \pi_i = k) \tag{4}$$

which is the probability of having emitted the first partial sequence up to i ending at state k.
The backward probability is:

$$b_k(i) = P(O_{i+1}, \ldots O_{L-1}, O_L|\pi_i = k) \tag{5}$$

which is the probability of having emitted the sequence starting from the last element back to the (i+1)th element, given that we end at position i in state k. The probability of emitting the whole sequence can be computed using either forward or backward according to:

$$P(O|M) = f_{END}(L+1) = b_{BEGIN}(0) \tag{6}$$

Forward and backward are also necessary for updating of the HMM parameters, using the Baum-Welch algorithm [3, 7]. Alternative a gradient-based training algorithm can be applied [3, 13].

Viterbi decoding

Viterbi decoding finds the path (π) through the model which has the maximal probability with respect to all the others [3, 7]. This means that we look for path which is

$$\pi^v = argmax_{\{\pi\}} P(\pi|O, M) \tag{7}$$

where $O(=O_1, \ldots O_L)$ is the observed sequence of length L and M is the trained HMM model. Since the $P(O|M)$ is independent of a particular path π, Equation 7 is equivalent to

$$\pi^v = argmax_{\{\pi\}} P(\pi, O|M) \tag{8}$$

$P(\pi, O|M)$ can be easily computed as

$$P(\pi, O|M) = \prod_{i=1}^{L} a_{\pi(i-1),\pi(i)} e_{\pi(i)}(O_i) \cdot a_{\pi(L),END} \tag{9}$$

where by construction $\pi(0)$ is always the $BEGIN$ state.

Defining $v_k(i)$ as the probability of the most likely path ending in state k at position i, and $p_i(k)$ as the trace-back pointer, π^v can be obtained running the following dynamic programming called Viterbi decoding

- **Initialization**

$$v_{BEGIN}(0) = 1 \quad v_k(0) = 0 \quad for \quad k \neq BEGIN$$

- **Recursion**

$$v_k(i) = [\max_{\{s\}}(v_s(i-1)a_{s,k})]e_k(O_i)$$
$$p_i(k) = argmax_{\{s\}}v_s(i-1)a_{s,k}$$

- **Termination**

$$P(O, \pi^v|M) = \max_{\{s\}}[v_s(L)a_{s,END}]$$
$$\pi_L^v = argmax_{\{s\}}[v_s(L)a_{s,END}]$$

- **Traceback**

$$\pi_{i-1}^v = p_i(\pi_i^v) \quad for \quad i = L \dots 1$$

- **Label assignment**

$$\Lambda_i = label(\pi_i^v) \quad for \quad i = 1 \dots L$$

1-best decoding

The 1-best labeling algorithm described here is the Krogh's previously described variant of the N-best decoding [13]. Since there is no exact algorithm for finding the most probable labeling, 1-best is an approximate algorithm which usually achieves good results in solving this task [13]. Differently from Viterbi, the 1-best algorithm ends when the most probable labeling is computed, so that no trace-back is needed.

For sake of clarity, here we present a redundant description, in which we define H_i as the set of all labeling hypothesis surviving as 1-best for each state s up to sequence position i. In the worst case the number of distinct labeling-hypothesis is equal to the number of states. h_i^s is the current partial labeling hypothesis associated to the state s from the beginning to the i-th sequence position. In general several states may share the same labeling hypothesis. Finally, we use $\oplus$ as the *string concatenation operator*, so that

’AAAA’$\oplus$’B’=’AAAAB’. 1-best algorithm can then described as

- **Initialization**

$$\begin{array}{rr} v_{BEGIN}(0) = 1 & v_k(0) = 0 \quad for \quad k \neq BEGIN \\ v_k(1) = a_{BEGIN,k} \cdot e_k(O_1) & H_1 = \{label(k) : a_{BEGIN,k} \neq 0\} \\ H_i = \emptyset & for \quad i = 2, \dots L \end{array}$$

- **Recursion**

$$\begin{array}{rc} v_k(i+1) = & \max_{h \in H_i}[\sum_s v_s(i) \cdot \delta(h_i^s, h) \cdot a_{s,k})]e_k(O_i) \\ h_{i+1}^k = & argmax_{h \in H_i}[\sum_s v_s(i) \cdot \delta(h_i^s, h) \cdot a_{s,k})] \oplus label(k) \\ H_{i+1} \leftarrow & H_{i+1} \quad \bigcup \quad \{h_{i+1}^k\} \end{array}$$

- **Termination**

$$\Lambda = argmax_{h \in H_L} \sum_s v_s(L) \delta(h_L^s, h) a_{s,END}$$

where we use the Kronecker’s delta $\delta(a, b)$ (which is 1 when $a = b$, 0 otherwise). With 1-best decoding we do not need keeping backtrace matrix since Λ is computed during the forward steps.

Posterior decoding

The *posterior* decoding finds the path which maximizes the product of the *posterior* probability of the states [3, 7]. Using the usual notation for forward ($f_k(i)$) and backward ($b_k(i)$) we have

$$P(\pi_i = k|O, M) = f_k(i)b_k(i)/P(O|M) \tag{10}$$

The path π^p which maximizes the posterior probability is then computed as

$$\pi_i^p = argmax_{\{s\}} P(\pi_i = s|O, M) \quad for \quad i = 1 \dots L \tag{11}$$

The corresponding label assignment is

$$\Lambda_i = label(\pi_i^p) \quad for \quad i = 1 \ldots L \tag{12}$$

If we have more than one state sharing the same label, labeling can be improved by summing over the states that share the same label (*posterior sum*). In this way we can have a path through the model which maximizes the posterior probability of being in a state with *label* λ when emitting the observed sequence element , or more formally:

$$\Lambda_i = argmax_{\{\lambda\}} \sum_{label(s)=\lambda} P(\pi_i = s|O, M) \quad for \quad i = 1 \ldots L \tag{13}$$

The posterior-decoding drawback is that the state path sequences π^p or Λ may be not allowed paths. However, this decoding can perform better than Viterbi, when more than one high probable path exits [3, 7]. In this case a post-processing algorithm that recast the original topological constraints is recommended [8].

In the sequel, if not differently indicated, with the term *posterior* we mean the posterior sum.

Posterior-Viterbi decoding

Posterior-Viterbi decoding is based on the combination of the Viterbi and posterior algorithms. After having computed the *posterior* probabilities we use a Viterbi algorithm to find the best *allowed posterior* path through the model. A related idea, specific for pairwise alignments was previously introduced to improve the sequence alignment accuracy [9].

In the PV algorithm, the basic idea is to compute the path π^{PV}

$$\pi^{PV} = argmax_{\{\pi \in A_p\}} \prod_{i=1}^{L} P(\pi_i|O, M) \tag{14}$$

where A_p is the set of the allowed paths through the model, and $P(\pi_i|O, M)$ is the *posterior* probability of the state assigned by the path π at position i (as computed in Eq. 10).

Defining a function $\delta^*(s,t)$ that is 1 if $s \rightarrow t$ is an allowed transition of the model M, 0 otherwise, $v_k(i)$ as the probability of the most probable *allowed-posterior* path ending at state k having observed the partial $O_1, \dots O_i$ and p_i as the trace-back pointer, we can compute the best path π^{PV} using the Viterbi algorithm

- **Initialization**

$$v_{BEGIN}(0) = 1 \quad v_k(0) = 0 \quad for \quad k \neq BEGIN$$

- **Recursion**

$$\begin{aligned} v_k(i) = \max_{\{s\}}[v_s(i-1)\delta^*(s,k)]P(\pi_i = k|O, M) \\ p_i(k) = argmax_{\{s\}}[v_s(i-1)\delta^*(s,k)] \end{aligned}$$

- **Termination**

$$\begin{aligned} P(\pi^{PV}|M, O) = max_s[v_s(L)\delta^*(s, END)] \\ \pi_L^{PV} = argmax_{\{s\}}[v_s(L)\delta^*(s, END)] \end{aligned}$$

- **Traceback**

$$\pi_{i-1}^{PV} = p_i(\pi_i^{PV}) \quad for \quad i = L \dots 1$$

- **Label assignment**

$$\Lambda_i = label(\pi_i^{PV}) \quad for \quad i = 1 \dots L$$

Datasets

Two different types of data are used to score the posterior-Viterbi algorithm, namely synthetic and real data. In the former case, we start with the simple *occasionally dishonest casino* illustrated in [7], referred here as *LF* model (Figure 1); then we increase the complexity of the automaton with other two models. First, we introduce the *occasionally dishonest casino* reported in Figure 2 and referred as $L2F2$, in which fair (label F) and loaded dice (label L) come always in pairs (or one die is always tossed twice). A third more complex version of the *occasionally dishonest casino* is shown in Figure 3 (model $L3F3$). In $L3F3$ the loaded dice are multiple of three (or one die is always tossed three times), while the number of fair tosses are at least three but can be more.

Accordingly, for each toy model presented above (Figures 1, 2 and 3), we produced 50 sequences of 300 dice outcomes and we trained the corresponding empty models (one for each models) using the Baum-Welch algorithm. The initial empty models have the same topology of the models LF, $L2F2$ and $L3F3$, with their emission and allowed transition probabilities set to the uniform distribution.

After training, we tested the ability of different algorithms (Viterbi, 1-best and PV) to recover the original *labeling* from the observed sequence of numbers (the dice outcomes).

The problem of the prediction of the all-beta transmembrane regions is used to test the algorithm on real data application. In this case we use a set that includes 20 constitutive beta-barrel membrane proteins whose sequences are less than 25% homologous and whose 3D structure have been resolved. The number of beta-strands forming the transmembrane barrel ranges from 2 to 22. Among the 20 proteins 15 were used to train a circular HMM (described in [16]), and here are tested in cross-validation (1a0sP, 1bxwA, 1e54, 1ek9A, 1fcpA, 1fep, 1i78A, 1k24, 1kmoA, 1prn, 1qd5A, 1qj8A, 2mprA, 2omf, 2por). Since there is no detectable sequence identity among the selected 15 proteins, we adopted a leave-one-out approach for training the HMM and testing it. All the reported results are obtained during the testing phase, and the complete set of results is available at www.biocomp.unibo.it/piero/posvit.

The other 5 new proteins (1mm4, 1nqf, 1p4t, 1uyn, 1t16) are used as a blind new test.

Measures of accuracy

We used three indices to score the accuracy of the algorithms. The first one is Q_2 which computes the number of correctly assigned labels divided by the total number of observed symbols. Then we use the SOV index [20] to evaluate the segment overlaps. Finally, in the case of the all-beta transmembrane proteins we adopt a very stringent measure called Q_{ok}: a prediction is considered correct *only if the number of transmembrane segments coincides with the observed one and the corresponding segments have a minimal overlap of m residues* [8]. The value m is segment-dependent and for each segment pairs, is computed as

$$m = min\{|seg_{pr}|/2, |seg_{ob}|/2\} \tag{15}$$

where $|seg_{pr}|$ and $|seg_{ob}|$ are the predicted and observed segment lengths, respectively.

Results and Discussion

Testing the decoding algorithms on toy models

We start using one of the simplest HMM model that can be thought of (LF), which is the *occasionally dishonest casino* presented in [7]. LF can parse any kind of observed sequence of numbers ranging from 1 to 6 (the die faces), generated with loaded and fair dice. Based on the LF model we produced 50 sequences with 300 dice outcomes and we trained an empty model with them. After this, we tested the three decoding algorithms that preserve the automaton grammar on the task of reconstructing the correct labeling.

In Table 1, we show that the accuracy of the posterior-Viterbi is greater than those of the other two algorithms. It is worth noticing that with this simple model the posterior algorithm alone achieves a similar accuracy (data not shown).

The $L2F2$ and $L3F3$ models, in which no one of the posterior decoding reconstructions is consistent with the automaton grammar (not parsable) are of some interest. In this case, among the three grammar-preserving algorithms, the posterior-Viterbi is the best performing one. This is particularly true for the $L3F3$ model, in which the SOV values highlight a quite good performance of PV. Considering that the three reconstructed models, as computed with the Baum-Welch, are very similar to the theoretical ones and independent from decoding, it is worth noticing the performance drop of the Viterbi algorithm. From these results it appears that in some cases the use of the PV decoding leads to a better performance given the same data and the same model.

Testing the decoding algorithms on real data

In order to test our decoding algorithm on real biological data, we used a previously developed HMM, devised for the prediction of the topology of beta-barrel membrane proteins [16]. The hidden Markov model is a sequence-profile-based HMM and takes advantage of emitting vectors instead of symbols, as described in [16].

Since the previously designed and trained HMM [16] emits profile vectors, sequence profiles have been computed from the alignments as derived with PSI-BLAST [1] on the non-redundant database of protein sequences (ftp://ftp.ncbi.nlm.nih.gov/blast/db/) .

The results obtained using the four different decoding algorithms are shown in Table 2, where the performance is tested with a jack-knife validation procedure for the first 15 proteins and as blind-test for the latter 5 (see Methods). It is evident that for the problem at hand the Viterbi decoding and the 1-best are unreliable, since only one of the proteins is correctly assigned. In this case the posterior decoding is more efficient and can correctly assign 60% and 40% of the proteins, in cross-validation and on the blind set, respectively. Here the posterior decoding is used without MaxSubSeq , introduced before to recast the grammar [16].

From Table 2 it evident that the new PV decoding is the best performing decoding achieving 80% and 60% accuracy in cross-validation and on the blind set, respectively. This is done ensuring that predictions are consistent with the designed automaton grammar.

Comparison with other available HMMs

Although this is out of the scope of this paper, the reader may be interested in seeing a comparison between our HMM-decoding with those obtained from the available web servers, based on similar approaches [2, 6]. In Table 3 we show the results. The **tmbb** server [2] allows the user to test three different algorithms, namely Viterbi, 1-best and posterior. Differently from us they find that their HMM does not show significant differences among the three decoding algorithms. This dissimilar behaviour may be due to several concurring facts: *i*) the different HMM models, *ii*) **tmbb** runs on a single-sequence input, *iii*) **tmbb** is trained using the *Conditional Maximum Likelihood* [12].

The second server **PROFtmb** [6] is based on a method that exploits multiple sequence information and posterior probabilities. Their decoding is related to the posterior-Viterbi; however, in their algorithm the authors first obtained the posterior sum contracted into two possible labeling (inner/outer loops and transmembrane as we did in [16]), then they made use of the explicit value of the HMM transition probabilities ($a_{i,j}$). In this way they count the transition probabilities twice (implicitly in the posterior-probability and directly into their algorithm) and the **PROFtmb** performance is not very different from ours. In our opinion, the fact that the newly implemented PV algorithm performs similarly or better, with respect to all indices, suggests that PV can be useful also when applied to the other HMM models.

Conclusions

The new PV decoding algorithm is more convenient in that overcomes the difficulties of introducing a problem-dependent optimization algorithm when the automaton grammar is to be re-cast. When one-state-path dominates we may expect that PV does not perform better than the other decoding algorithms, and in these cases the 1-best is preferred [13]. Nevertheless, we show that when several concurring paths are present, as in the case of our beta-barrel HMM, PV performs better than the others.

A performance similar to that obtained with PV decoding can be achieved using MaxSubSeq algorithm [8] on top of the posterior sum decoding. However, although MaxSubSeq is a very general two-class segment optimization algorithm, PV is far more useful when the underlying predictor is a HMM, where more than two labels and different constraints can be introduced into the automaton grammars.

Although PV takes a time longer than other algorithms (the posterior + the Viterbi time), the PV asymptotic computational time-complexity still remains, as for the other decodings $O(N^2 \cdot L)$ (where L and N are the protein length and the number of states, respectively). As far as the memory requirement is concerned, PV needs the same space-complexity of the Viterbi and posterior ($O(N \cdot L)$), while 1-*best* in the average case requires less memory, and can also be reduced [13]. When computational speed is an issue, Viterbi algorithm is the fastest and the time complexity order is $time(viterbi) \leq time(1 - best) \leq time(PV)$.

Finally, PV satisfies any HMM grammar structures, including automata containing silent states, and it is applicable to all the possible HMM models with an arbitrary number of labels and without having to work out a problem-dependent optimization algorithm.

List of abbreviations

- **HMM**: hidden Markov model.
- **PV**: Posterior-Viterbi.

Authors' contributions

PF developed the Posterior-Viterbi algorithm. PLM designed and trained the Hidden Markov Models. RC contributed to the problem. PF, PLM and RC authored the manuscript.

Acknowledgements

This work was partially supported by the BioSapiens Network of Excellence, a grant of the Ministero della Università e della Ricerca Scientifica e Tecnologica (MURST), a grant for a target project in Biotechnology (CNR), a project on Molecular Genetics (CNR), a PRIN 2002 and a PNR 2001-2003 (FIRB art.8).

References

[1] Altschul,S.F., Madden,T.L., Schaffer,A.A., Zhang,J., Zhang,Z., Miller,W. and Lipman,D.J. (1997) Gapped BLAST and PSI-BLAST: A new generation of protein database search programs, Nucleic Acid Res., 25, 3389-3402.

[2] Bagos,P.G., Liakopoulos,T.D., Spyropoulos,I.C., Hamodrakas,S.J. (2004) PRED-TMBB: a web server for predicting the topology of beta-barrel outer membrane proteins, Nucleic Acids Res., 32W400-W404.

[3] Baldi,P. and Brunak,S. (2001) Bioinformatics: the Machine Learning Approach MIT Press.

[4] Baldi,P., Chauvin,Y., Hunkapiller,T., and McClure,M.A. (1994) Hidden Markov Models of Biological Primary Sequence Information, PNAS USA, 91, 1059-1063.

[5] Bateman,A., Birney,E., Cerruti,L., Durbin,R., Etwiller,L., Eddy,S.R., Griffiths-Jones,S., Howe,K.L., Marshall,M. and Sonnhammer,E.L. (2002) The Pfam Protein Families Database, Nucleic Acids Research, 30,276-280.

[6] Bigelow,H.R., Petrey,D.S., Liu,J., Przybylski,D., and Rost,B. (2004) Predicting transmembrane beta-barrels in proteomes, Nucleic Acids Res., 32, 2566-2577.

[7] Durbin,R., Eddy,S., Krogh,A. and Mitchinson,G. (1998) Biological sequence analysis: probabilistic models of proteins and nucleic acids. Cambridge Univ. Press, Cambridge.

[8] Fariselli P., Finelli,M., Marchignoli,D., Martelli,P.L., Rossi,I. and Casadio,R. (2003) MaxSubSeq: an algorithm for segment-length optimization. The case study of the transmembrane spanning segments, Bioinformatics 19,500-505.

[9] Holmes,I., and Durbin,R. (1998) Dynamic programming alignment accuracy, J Comput Biol., 493-504.

[10] Liu,Q., Zhu,Y.S., Wang,B.H., and Li,Y.X (2003) A HMM-based method to predict the transmembrane regions of beta-barrel membrane proteins. Comput Biol Chem. 27,69-76.

[11] Krogh,A., Brown,M., Mian,I.S., Sjolander,K., and Haussler,D. (1994) Hidden Markov models in computational biology: Applications to protein modeling, Journal of Molecular Biology, 235,1501-1531.

[12] Krogh,A. (1994) Hidden Markov models for labeled sequences. In Proceedings 12th International Conference on Pattern Recognition. IEEE Comp. Soc. Press, Singapore, pp.140-144.

[13] Krogh,A. (1997) Two methods for improving performance of a HMM and their application for gene finding. Proceedings of the Fifth International Conference on Intelligent Systems for Molecular Biology, pages 179-186, Menlo Park, CA, AAAI Press

[14] Krogh,A., Larsson,B., von Heijne,G. and Sonnhammer,EL. (2001) Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes, J. Mol. Biol., 305, 567-580.

[15] Mamitsuka,H. (1998) Predicting peptides that bind to MHC molecules using supervised learning of hidden Markov models, Proteins 33,460-474.

[16] Martelli,P.L., Fariselli,P., Krogh,A., Casadio,R. (2002) A sequence-profile-based HMM for predicting and discriminating beta barrel membrane proteins, Bioinformatics 18, S46-S53.

[17] Martelli,P.L., Fariselli,P., and Casadio,R. (2003) An ENSEMBLE machine learning approach for the prediction of all-alpha membrane proteins, Bioinformatics, 19,i205-i211.

[18] Tusnady,G.E. and Simon,I. (1998) Principles governing amino acid composition of integral membrane proteins: application to topology prediction, J. Mol. Biol., 283, 489-506.

[19] Viklund,H., and Elofsson,A. (2004) Best alpha-helical transmembrane protein topology predictions are achieved using hidden Markov models and evolutionary information. Protein Sci., 13,1908-1917.

[20] Zemla,A., Venclovas,C., Fidelis,K., Rost,B. (1999) A modified definition of Sov, a segment-based measure for protein secondary structure prediction assessment. Proteins, 34,220-223.

Table 1: Accuracy of the different algorithms on the toy-models

Algorithms	toy-models		
	LF	L2F2	L3F3
viterbi			
Q2	0.80	0.86	0.47
SOV	0.48	0.73	0.35
SOV(L)	0.42	0.64	0.37
1-best			
Q2	0.80	0.86	0.88
SOV	0.48	0.73	0.81
SOV(L)	0.42	0.64	0.72
posterior-Viterbi			
Q2	0.82	0.88	0.90
SOV	0.66	0.80	0.82
SOV(L)	0.61	0.75	0.78

For the indices see 'Measure of accuracy' section. SOV(L)= SOV computed for the loaded class only.

Table 2: Q_{ok} prediction accuracy obtained with the four different decoding algorithms on the real data

Proteins	Viterbi	1-best	posterior	posterior-Viterbi
cross-validation				
1a0spTOT	-	-	-	OK
1bxwaTOT	-	-	OK	OK
1e54	-	-	OK	OK
1ek9aTOT	-	-	OK	OK
1fcpaTOT	-	-	-	-
1fepTOT	-	-	-	OK
1i78a	-	-	OK	OK
1k24	-	-	-	OK
1kmoaTOT	-	-	OK	OK
1prn	-	-	-	-
1qd5a	-	-	OK	OK
1qj8a	-	-	OK	OK
2mpra	-	-	OK	OK
2omf	-	-	OK	OK
2por	-	-	-	-
$< Q_{ok} >$	0.0	0.0	0.60	0.80
blind-test				
1mm4	-	-	OK	-
1nqf	-	-	-	OK
1p4t	OK	OK	OK	OK
1uyn	-	-	-	OK
1t16	-	-	-	-
$< Q_{ok} >$	0.20	0.20	0.40	0.60

$Q_{ok} >$: see Measures of Accuracy.

Table 3: Posterior-Viterbi accuracy compared with other algorithms and HMM models

Method	Q2	SOV	SOV(BetaTM)	SOV(Loop)	Q_{ok}
cross-validation[4]					
Posterior-Viterbi[1]	0.82	0.87	0.92	0.81	0.80
Viterbi[1]	0.63	0.33	0.27	0.35	0.0
1-best[1]	0.65	0.37	0.31	0.38	0.0
PROFTmb[2]	0.83	0.87	0.88	0.84	0.73
tmbb[3] (Viterbi)	0.78	0.83	0.81	0.82	0.60
tmbb[3] (1-best)	0.78	0.83	0.81	0.82	0.60
tmbb[3] (posterior)	0.78	0.82	0.80	0.82	0.60
blind-test[4]					
Posterior-Viterbi[1]	0.80	0.81	0.84	0.74	0.60
Viterbi[1]	0.62	0.38	0.35	0.40	0.20
1-best[1]	0.63	0.38	0.36	0.40	0.20
PROFTmb[2]	0.72	0.65	0.72	0.58	0.40
tmbb[3] (Viterbi)	0.71	0.73	0.79	0.71	0.20
tmbb[3] (1-best)	0.71	0.73	0.79	0.71	0.20
tmbb[3] (posterior)	0.72	0.75	0.81	0.71	0.20

1 Model taken from Martelli et al., 2002 [16]

2 Bigelow et al., (2004) [6]

3 Bagos et al., 2004 [2]

4 this is only referred to posterior-Viterbi decoding

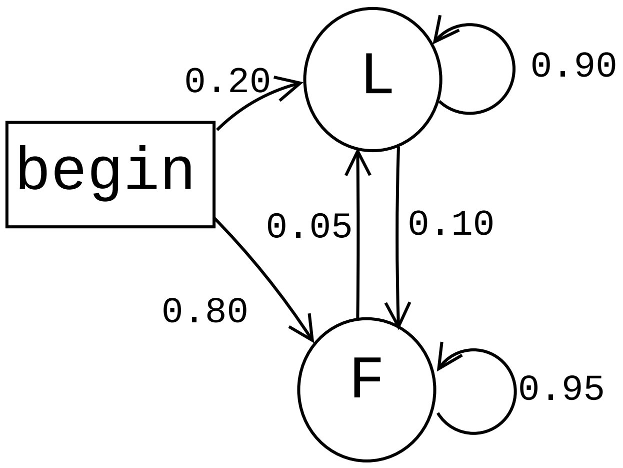

Figure 1: Occasionally dishonest casino (Model LF). The emission probabilities of the fair state (F) are 1/6 for each possible outcome, while in the loaded die the emission probabilities are 1/2 for the '1' and 1/10 for the other faces.

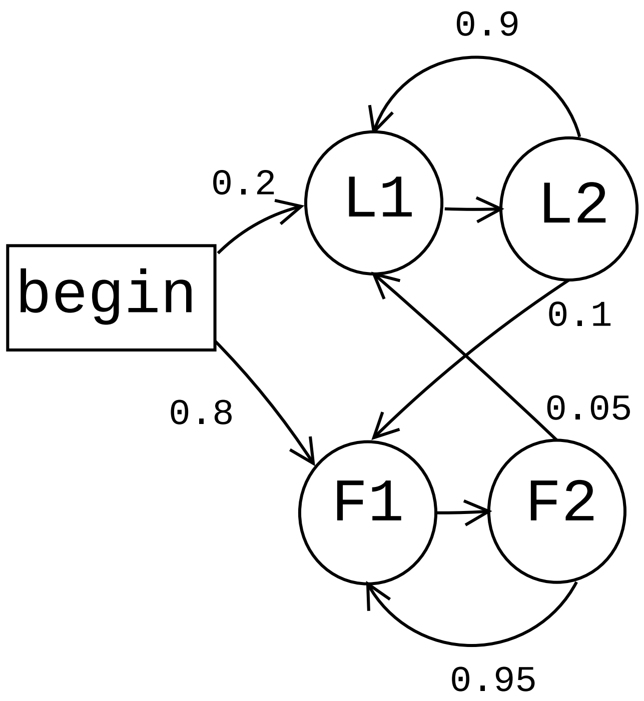

Figure 2: Occasionally dishonest casino (Model L2F2). For the emission probabilities see Figure 1.

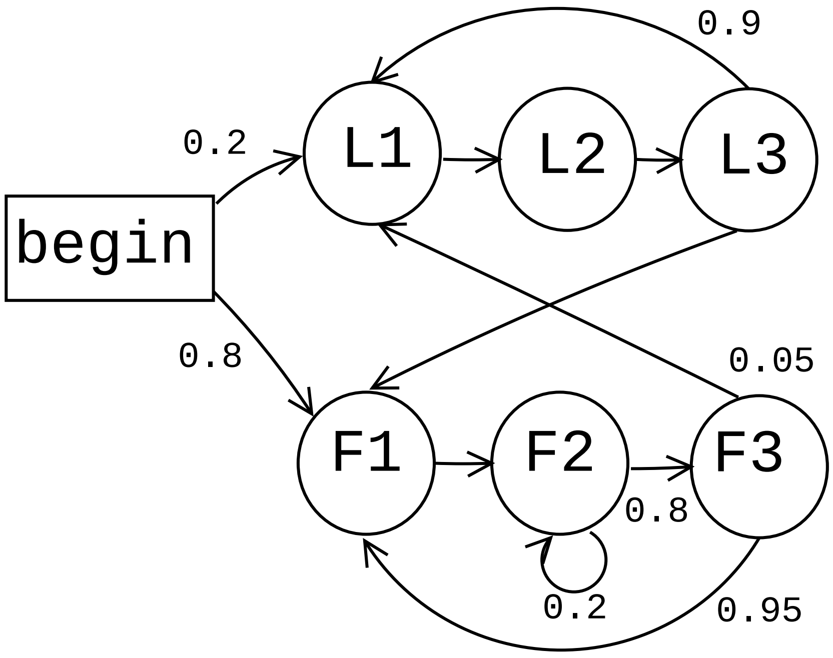

Figure 3: Occasionally dishonest casino (Model L3F3). For the emission probabilities see Figure 1.